CMPT417/CMPT812
Department of Computer Science
University of Saskatchewan
December 16/19, 1995
9:00am
Three Hours

Read all pages of this exam before starting. There are seven questions. Do four of them. All questions have equal weight.

This exam is open book. Answer all questions in the exam booklets provided.

1. Below is a "vanilla" meta interpreter.

```
prove(true) :- !.
prove((X,Y)) :- !,
        prove(X),
        prove(Y).
prove(G) :-
        rule((G:-B)),          % (**)
        prove(B).

rule((a(X):-b(X))).
rule((a(X):-c(X),X is 3*3)).
rule((c(X):-true)).
```

Note that I have used the "rule" representation we used in the latter half of the course, rather than PROLOG's built in clause metapredicate. The above meta-interpreter will fail on the query

```
|?- prove(a(X)).
```

because it does not understand the "is" metapredicate.

Rewrite the above interpreter so that

a) it can access PROLOG's built in arithmetic via the "is" clause, and,
b) it prints a message of the form

```
<CLAUSE-NAME> : no such rule
```

whenever it cannot find a rule to match G in the line (**).

For example, your meta-interpreter should perform as follows:

```
| ?- prove(a(X)).
b(X): no such rule

X = 9;
no
| ?- prove(X is 3*4+5).

X = 17 ? ;

no
| ?- prove(b(X)).
b(X): no such rule

no
| ?-
```

(Hint: it is important that your meta-interpreter not succeed after it discovers that no rule matches some clause.)

2. Below is a depth-bounded PROLOG meta-interpreter, that, given a goal and a maximum depth, it returns the depth at which a proof was found.

```
%
% prove(G,CurrDepth,FinalDepth,MaxDepth) means
%     try to prove the goal G, going to a depth
%     of no more than MaxDepth.  CurrDepth is the current
%     depth of the search, FinalDepth is the answer.

% terminate the search if gone too deep

prove(_,D,_,M) :- D > M, !, fail.

prove(true,D,D,_) :- !.

prove((X,Y),D0,D3,M) :- !,
        prove(X,D0,D1,M),
        prove(Y,D0,D2,M),
        max(D1,D2,D3).

prove(G,D1,D3,M) :-
        rule((G:-B)).
        D2 is D1 + 1,
        prove(B,D2,D3,M).

max(X,Y,X) :- X > Y.
max(X,Y,Y) :- X =< Y.
```

Suppose you add the database:

```
rule((a :- b,m)).              %% a :- b,m.
rule((b :- c)).                %% b :- c.
rule((c :- d)).                %% c :- d.
rule((d :- e)).                %% d :- e.
rule((e :- true)).             %% e.
rule((m :- true)).             %% m :- n.
rule((a :- n)).                %% a.
rule((n :- true)).             %% n.
```

a) What answers do you get for the following queries? (Give ALL answers to EACH query.)

```
i)      |?- prove(a,0,Ans,5).
ii)     |?- prove(a,0,Ans,3).
iii)    |?- prove(a,0,Ans,1).
```

if the user wants to search to a depth of no more than 100 steps.

b) Rewrite the meta-interpreter so that INSTEAD of restricting the maximum depth, it restricts maximum number of inferences. To say the same thing another way, it restricts the total number of times the "rule" clause is used. Each answer returns the total "length" of that proof.

For example, my solution gives the following answers on backtracking:

```
| ?- prove(a,0,D,10).

D = 6 ? ;
D = 2 ? ;
no
```

(Hint: A correct answer could be 3 lines shorter.)

3. Consider the following logic program:

```
a,b => c.          (1)
d,m => a.          (2)
e => a.            (3)
f,m => b.          (4)
g => b.            (5)
h => b.            (6)
d or e.            (7)
f or g or h.       (8)
m.                 (9)
```

a) How many proofs of "c" are there using a meta interpreter with rule NA?

b) How many proofs of "c" succeed if you use the method of ordered clauses?

For (a) and (b) above, first draw the entire AND-OR tree. Circle successful nodes (proved because they are facts) and mark successful branches with a check mark and "NA" if you use the negated ancestor rule. For part (a) give a combinatorial argument based on the number of OR-choices. For part (b), use the clause numbering provided to show which branches don't succeed. ASSUME THAT THE NUMBER AT THE LOWER NODE MUST BE LOWER THAN THE NUMBER AT THE HIGHER NODE FOR THE BRANCH TO SUCCEED.

4. The name of this class is "knowledge representation and reasoning". During the discussion of inheritance hierarchies, we observed that the choice of representation may greatly facilitate reasoning. This may not be much different from saying that the choice of data structure may greatly facilitate an algorithm.

In class, we discussed three different approaches to a problem called "diagnosis".

a) What is "diagnosis"?

b) The three different approaches to diagnosis required different kinds of knowledge and used different patterns of reasoning. Briefly describe each.

c) Imagine that you are a diagnostician in some "real world" application, something significantly more complex than the "hay fever-sneeze" or "three gate" problems we have discussed. Consider the following problem. Imagine that you have put considerable effort into constructing a diagnostic theory and that this theory is confirmed by a vast body of evidence. This evidence could be hundreds of carefully supervised experiments, etc. You have also constructed a diagnostic computer program, along the lines of those studied in class to assist you.

You now receive a report from your laboratory that some new evidence completely contradicts the hypothesis you have carefully constructed. Your diagnostic computer program confirms this.

The diagnostic engines we discussed all collapse at this point. But real diagnosticians will consider the possibility that the evidence may be wrong. For example, an error in the evidence may be explained by anything from a typographical error to a measurement error to a freak of nature.

Discuss this problem from the perspective of automated diagnosis. In some sense, this is the problem of unreliable evidence. In the context of knowledge representation and reasoning, discuss how automated diagnosis might still work, how you might troubleshoot unreliable evidence, and how you might decide, at appropriate points in the diagnostic procedure when to reject the theory and when to reject the evidence.

5. Consider the following default reasoning problem.

```
frog(X) => amphibian(X).
salamander(X) => amphibian(X).
amphibian(X) and d1(X) => n(hop(X)).
frog(X) => hop(X).
amphibian(X) and d2(X) => n(tail(X)).
salamander(X) => tail(X).
amphibian(X) => frog(X) or salamander(X).

frog(fran).
salamander(sal).
amphibian(arnie).

default(d1(X)).      %% normal amphibs don't hop
default(d2(X)).      %% normal ambphibs don't have tails
```

NOTE: to simplify your proofs, you can use first letters for all the predicates other than the defaults. For the proofs, you don't need to show the unifications explicitly.

Using the method of default reasoning with negated ancestor rule, AND checking consistency of new defaults with rules and defaults so far, give A SINGLE ANSWER to each of the following queries. Show ONE proof tree for each of these, and give the answer your default reasoner would give. In the event there is no proof, indicate failures on your proof tree.

a) `|?- explain(tail(sal),E).`
b) `|?- explain(n(hop(fran)),E).`
c) `|?- explain(frog(arnie),E).`
d) `|?- explain(tail(arnie),E).`

Recall also the method given for proofs using default rules: The "proof by assumption" of a default is indicated by drawing a wavy line and attempting to produce a proof of the negation of the default from the rules and defaults USED SO FAR, below the wavy line. If the proof below the wavy line FAILS, the default above it SUCCEEDS, and vice-versa.

Without necessarily drawing any proof tree, what answer would the default reasoner give to the following query, and why?

e) `|?- explain((frog(arnie),tail(arnie)),E).`

6. Representation and reasoning.

The following contain stories and expressions and ask you to solve problems of representation and reasoning with each.

a) "Honest men are noble, but not all noble men are honest."

(apropos "Rob Roy")

Is the above consistent?

b) i) Assume you have predicates defined as follows:

built: means the computer center will be built
ct(X): means X contributes time
cl(X): means X contributes leadership
eq(X,Y): means X is equal to Y

Convert to clausal form:

"If everyone contributes time and someone contributes leadership, the new computer centre will be built.

Everyone other than John contributed time.

John may or may not have contributed time.

Milton contributed leadership.

The computer centre was not built."

Assume you have an "oracle" clause "eq(X,Y)" which always succeeds when X is equal to Y. By oracle clause, it is meant that the clause is smart enough to know that

batman=bruce_wayne

or

evening_star = morning_star

that is, when an object has different names, it knows when the names refer to the same object.

ii) Prove, using resolution, that someone did not contribute time.

iii) Using pure and simple resolution, you will not be above to prove that it was John that did not contribute time. Either explain why this is not possible, OR
show that it IS possible to prove

someone did not contribute time and that person was John.

c) Represent the following:

"University vehicles are not generally rented to students, with the exception of university vans. However, the extra large vans are reserved for Geology professors and are not rented to students."

The above involves a hierarchy of defaults. Your answer should include the obvious implicit knowledge and show
that you know how to use cancellation axioms to avoid incorrect answers.

7. a) "Migraines are headaches, and so are hemicrania and cephalagia.
Headache symptoms include loss of sleep and irritability.
Migraines are severe and can occur during the day or night.
Hemicrania is severe, but brief and generally occurs at night,
while cephalagia is a dull, persistent ache.
When cephalagia and migraines occur together, the pain is not
always severe, but nosebleed are frequently observed."

The above does not represent true medical knowledge.
Represent the above for use in your default reasoner.

Note that certain symptoms can be explained by very general or very specific diagnoses. For example, loss of sleep can be explained by the simpler explanation "headache" or the more specific explanation "migraine". Suggest a mechanism for handling this.

b) Suggest a mechanism for handling irrelevant symptoms. For example, consider the diagnostic database:

```
rule((sneeze:-cold)).
default(cold).
```

The query

```
|?- explain((sneeze,temperature),E).
```

will fail because temperature is irrelevant to this database and cannot be explained. Suggest a mechanism to cope with irrelevant information.